Testing Like A Boss

Arkady Galyash

TosChart

April 29, 2014



Agenda

1. How regular JUnit test looks like (in time)?

2. Links



└ Intro

Intro

- John Doe The Programmer
- Java developer @ Moon Ms
- Since 2000
- Binary search algorithm



└─ June 2000

June 2000



Just graduated, "tests are for chicken"



└ May 2001

May 2001



JUnit discovered



└ July 2006

July 2006



User tries to search non-existant element



December 2007

December 2007



User tries to search in array with duplicates



-How regular JUnit test looks like (in time)?

Cottober 2009

October 2009



Integer overflow



Summary

Long way











2000

2001

2006

2007

2009



□Summary

xUnit

- We need to think about test cases.
- Until we've discover one, we cannot find a bug
- If we discover a test case, this test is useless right now, it may be useful only in future



QuickCheck

QuickCheck

- Another approach
- You specify pre- and post-conditions
- Let computer generate input data



L QuickCheck

QuickCheck History

- Initially written for Haskell in 1999
- Re-implementations of QuickCheck exist for Java, Scala, Clojure, Groovy, ...



└ QuickCheck

QuickCheck implementations

```
JCheck
                              rubycheck
       cluckcheck
cl-quickcheck
                          QuickSmash
                        node-quickcheck
|cjunit-quickcheck
|simple-check
     QuickCheck 7
Scheme-Check
                           factcheck
```



QuickCheck

QuickCheck implementations





└─How regular JUnit test looks like (in time)?

└─ QuickCheck





└─ How regular JUnit test looks like (in time)?

└─ QuickCheck

Demo





Links

Introduction to QuickCheck2



Thank You!

